Low Bitrate Speech Codec
PC Simulation Program

Copyright © 2004-2009 Signalogic, Inc.

Revision A7

Dec 2009

Signalogic, Inc.
9617 Wendell @ Skillman
Dallas, TX 75243
Tel: 214-349-5551
Fax: 214-343-0163

http://www.signalogic.com
tech_support@signalogic.com

Revision History

Revision	Date	Changes	Author
A3	03/15/05	Updating document for 600bps.	MM
A4	06/02/09	Merging executables to combine bit-rates, updating documentation.	AA
A5	06/24/09	Updating document to include 4000bps	AA
A6	11/09/09	Updated document to include CVSD codec for 16Kbps. Renamed document.	AA
A7	12/21/09	Updated document to include G726 codec	AA

Table of Contents

Revision History **2**

Table of Contents **3**

1 Overview **4**

2 Latency Notes **5**

3 Install Instructions **6**

4 Run Instructions **7**

4.1 MELPe Command-Line Examples 10
4.2 CVSD Command-Line Examples 11
4.3 G726 Command-Line Examples 12

5 Input and Output File Format **13**

6 Intermediate Bitstream File Format **14**

6.1 MELPe 600 bps Bitstream File Format 14
6.2 MELPe 600 bps Bitstream File Format for 135 msec latency codec 15

7 Framing and Timing **17**

8 G726 Bit Exactness Procedure **18**

8.1 Procedure to test ITU G726 test vectors bit exactness 18

9 Packet Bit Allocation **20**

9.1 MELPe Packet Bit Allocation for 600 bps for 150 msec latency 20
9.2 MELPe Packet Bit Allocation for 600 bps for 135 msec latency 20

1 Overview

The PC simulation implements the following codecs :

- MELPe 2400, 1200, 600 bps
- MELPe-Plus 4000, 2400, 1200, 600 bps
- CVSD 32, 16 kbps
- G726 16 kbps

as a non-real-time fixed-point simulation that runs as a Win32 console application on a PC under Win9x, Win2k, or WinXP. The fixed-point simulation program was created as a Microsoft Visual Studio project; all source code files are in C.

2 Latency Notes

Below are latency notes for different bitrates and algorithm versions.

1. For MELPe 4000 bps, algorithm latency is 20 msec (160 speech samples at 8kHz sampling rate).
2. For MELPe 2400 bps, algorithm latency is 22.5 msec (180 speech samples at 8 kHz sampling rate).
3. For MELPe 1200 bps, algorithm latency is 67.5 msec (540 speech samples at 8 kHz sampling rate).
4. For MELPe 600 bps, algorithm latency is either 150 msec (1200 samples at 8 kHz sampling rate) or 135 msec (1080 samples at 8 kHz sampling rate).
5. For CVSD 16 kbps, algorithm latency is 4 samples
6. For CVSD 32 kbps, algorithm latency is 2 samples
7. For G726 16 kbps, algorithm latency is 0 samples.

   Note: The algorithm latency for CVSD is 4 or 2 samples because it is not a frame based algorithm. The implementation of the simulation program works on frames of speech for all codecs. As the framesize can be specified at the command-line as variable, you may want to consider final CVSD latency as dependent on the framesize.

3 Install Instructions

Below are Low Bitrate (LBR) Codec PC simulation program install instructions.

1. Unzip the file Low_Bitrate_Codec_Sim_Prog.zip.
2. After expanding the zip file, the following .exe files should be created:

File	Description
testcodec.exe	LBR codecs simulation executable
MelpE.dll, MelpEf.dll, Melp.dll, mcofile.dll, melp2forig.dll, melp40.dll, CVSD.dll, g726.dll	supporting DLLs
LBR_Codec_Sim_Prog_InstructionsRevA7.pdf	this document

See section 4 below for Run Instructions.

4 Run Instructions

Below are instructions to use the LBR Codec PC simulation program

1. To create encoded/decoded time domain files (e.g. .wav files), the LBR Codec simulation program takes a command-line of the form:

```
testcodec  encode_options  input_filename  decode_options  output_filename
```

Table 1 below shows a detailed description of command line parameters for the .exe file:

Encode Options		**Encode File**	**Decode Options**		**Decode File**
e | m1, m2, mf, cv, g76 | 06,12,20,24,40 | [t] | [np] | [mu, al] | [fnnn]		**Input Filename**	**d | m1, m2, mf, cv | 06,12,20,24,40 | [t] | [pf] | [mu, al] | [fnnn]**		**Output Filename**
e	Encode	sample_input.wav	d	Decode	sample_output.wav
	Codec			Codec	
m1	MELP		m1	MELP	
m2	MELPe		m2	MELPe	
mf	floating-point MELPe		mf	floating-point MELPe	
mt	transcode		mt	transcode	
bp	BYPASS		bp	BYPASS	
cv	CVSD		cv	CVSD	
g76	G726		g76	G726	
	Bitrate (bps)			Bitrate (bps)	
06	600 (150 msec latency)		06	600 (150 msec latency)	
06b	600 (135 msec latency)		06b	600 (135 msec latency)	
12	1200		12	1200	
20	2000		20	2000	
24	2400		24	2400	
40	4000		40	4000	
16	16000 (CVSD and G726)		16	16000 (CVSD and G726)	
32	32000 (CVSD only)		32	32000 (CVSD only)	
t	Transcode up/down		t	Transcode up/down	
	Noise Pre-Processor			Post-Filter	
	off			off	
np	on		pf	on	
	Framesize (samples)			Framesize (samples)	
f200	200(only with 600bps)		f200	200(only with 600bps)	
f180	180		f180	180	
	etc			etc	
mu	μ - law		mu	μ - law	
al	a – law		al	a – law	
sr	input amplitude limitation to 14 bit (with G726 only)		sr	input amplitude limitation to 14 bit (with G726 only)	

Table 1 - Command-Line Parameters for Output Encoded/Decoded Speech File Generation

Note : For some combination of options, error messages may be displayed depending on which .exe file you have. Please feel free to contact Signalogic in case this happens and we can send you the correct .exe file if you let us know the error.

2. To produce bitstream files, the LBR Codec simulation program accepts a command-line of the form:

```
testcodec  encode_options input_file  decode_options  bitstream_file
```

Table 2 below shows a detailed description of command line parameters for the .exe file:

Encode Options		**Encode File**	**Decode options**		**Bitstream File**
e | m1, m2, mf, cv | 06,12,20,24,40,16,32 | [t] | [np] | [fnnn] | eb		**Input Filename**	**d | m1, m2, mf, cv | 06,12,20,24,40,16,32 | [t] | [np] | [fnn] | db**		**Bitstream Filename**
e	Encode	sample_input.wav	d	Decode	sample_output.bit
	Codec			Codec	
m1	MELP		m1	MELP	
m2	MELPe		m2	MELPe	
mf	floating-point MELPe		mf	floating-point MELPe	
mt	transcode		mt	transcode	
bp	BYPASS		bp	BYPASS	
cv	CVSD		cv	CVSD	
	Bitrate (bps)			Bitrate (bps)	
06	600 (150 msec latency)		06	600 (150 msec latency)	
06b	600 (135 msec latency)		06b	600 (135 msec latency)	
12	1200		12	1200	
20	2000		20	2000	
24	2400		24	2400	
40	4000		40	4000	
16	16000 (CVSD only)		16	16000 (CVSD only)	
32	32000 (CVSD only)		32	32000 (CVSD only)	
	Noise Pre-Processor			Noise Pre-Processor	
	off			off	
np	on		np	on	
	Framesize (samples)			Framesize (samples)	
f200	200(only with 600bps)		f200	200(only with 600bps)	
f180	180		f180	180	
	etc			etc	
eb	Encode bitstream		db	Decode bitstream	

Table 2 - Command-Line Parameters for Bitstream File Generation

4.1 MELPe Command-Line Examples

A typical program command line for MELPe simulation would read as follows:

```
C:\> testcodec em224npf180 3.wav dm224pff180 3out.wav
```

The above example would implement the 2400 bps MELPe algorithm at a frame size of 180 samples. The noise pre-processor and the post filter are turned on during encoding and decoding, respectively. The file 3.wav (input waveform file) would be analyzed and synthesized (encoded and decoded, both) and output would be stored in the 3out.wav file.

The example below would implement the 4000 bps MELPe algorithm at a frame size of 160 samples. The noise pre-processor and the post filter are turned off during encoding and decoding, respectively. The file sample1.wav (input waveform file) would be analyzed and synthesized (encoded and decoded, both) and output would be stored in the sampleout.wav file.

```
C:\> testcodec em240f160 sample1.wav dm240f160 sampleout.wav
```

The example below would implement the 2400 bps MELPe algorithm at a frame size of 180 samples. The file Sample_input.wav (input waveform file) would be analyzed and **encoded** only and the bitstream would be stored in the MELPe_Plus_2400_Sample_bitstream file.

```
C:\> testcodec em224f180eb Sample_input.wav dm224f180db MELPe_Plus_2400_Sample_bitstream
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 200 samples. The noise pre-processor and the post filter are turned on during encoding and decoding, respectively. The file Sample_input.wav (input waveform file) would be analyzed and synthesized (encoded and decoded, both) and output would be stored in the MELPe_Plus_600_Sample_output.wav file.

```
C:\> testcodec eme06npf200 Sample_input.wav dme06pff200 MELPe_Plus_600_Sample_output.wav
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 200 samples. The noise pre-processor and the post filter are turned on during encoding and decoding, respectively. The file Sample_input.wav (input waveform file) would be analyzed and encoded only and the bitstream would be stored in the MELPe_Plus_600_Sample_bitstream file.

```
C:\> testcodec eme06npf200 Sample_input.wav pkg MELPe_Plus_600_Sample_bitstream
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 200 samples. The noise pre-processor and the post filter are turned on during encoding and decoding, respectively. The bitstream file MELPe_Plus_600_Sample_bitstream file (input bitstream file) would be analyzed and decoded only and the output would be stored in the MELPe_Plus_600_Sample_output.wav file.

```
C:\> testcodec pkg MELPe_Plus_600_Sample_bitstream dme06pff200 MELPe_Plus_600_Sample_output.wav
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 200 samples. The noise pre-processor and the post filter are turned off during encoding and decoding, respectively. The file Sample_input.wav (input waveform file) would be analyzed and synthesized (encoded and decoded, both) and output would be stored in the MELPe_Plus_600_Sample_output.wav file.

```
C:\> testcodec eme06f200 Sample_input.wav dme06f200 MELPe_Plus_600_Sample_output.wav
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 180 samples. The noise pre-processor and the post filter are turned off during encoding and decoding, respectively. The file Sample_input.wav (input waveform file) would be analyzed and synthesized (encoded and decoded, both) and output would be stored in the MELPe_Plus_600_Sample_output.wav file.

```
C:\> testcodec eme06f180 Sample_input.wav dme06f180 MELPe_Plus_600_Sample_output.wav
```

The example below would implement the 600 bps MELPe algorithm at a frame size of 180 samples. The noise pre-processor and the post filter are turned off during encoding and decoding, respectively. The file Sample_input.wav (input waveform file) would be analyzed and encoded only and the bitstream would be stored in the MELPe_Plus_600_Sample_bitstream file.

```
C:\> testcodec eme06f180 Sample_input.wav pkg MELPe_Plus_600_Sample_bitstream
```

4.2 CVSD Command-Line Examples

The example below would implement the 16 kbps CVSD algorithm taking speech in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and CVSD output would be stored in the sampleout.wav file.

```
C:\> testcodec ecv16f160 sample1.wav dcv16f160 sampleout.wav
```

The example below would implement the 32 kbps CVSD algorithm taking speech in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and CVSD output would be stored in the sampleout.wav file.

```
C:\> testcodec ecv32f160 sample1.wav dcv32f160 sampleout.wav
```

The example below would implement the 16 kbps CVSD algorithm taking speech in frames of 120 samples. The file sample1.wav (input waveform file) would be analyzed and CVSD output would be stored in the sampleout.wav file.

```
C:\> testcodec ecv16f120 sample1.wav dcv16f120 sampleout.wav
```

Notes :

1. The framesize for CVSD should be a multiple of 4 due to the nature of CVSD algorithm

2. The range of framesizes accepted by the Simulation program is 100 - 500 samples.

4.3 G726 Command-Line Examples

The example below would implement the 16 kbps G726 algorithm taking speech (16-bit linear format) in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and G726 output would be stored in the sampleout.wav file.

```
C:\> testcodec eg7616f160 sample1.wav dg7616f160 sampleout.wav
```

The example below would implement the 16 kbps G726 algorithm taking amplitude limited speech (14-bit linear) in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and G726 14-bit linear output would be stored in the sampleout.wav file.

```
C:\> testcodec eg7616srf160 sample1.wav dg7616srf160 sampleout.wav
```

The example below would implement the 16 kbps G726 algorithm taking μ-law input (usually used to test bit exactness of ITU test vectors) in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and G726 μ-law output would be stored in the sampleout.wav file.

```
C:\> testcodec eg7616muf160 sample1.wav dg7616muf160 sampleout.wav
```

Note : In this case, the μ-law values are stored as 16-bit integers in the wave file though the value of each sample is not greater than 255 (8-bit μ-law).

The example below would implement the 16 kbps G726 algorithm taking A-law input (usually used to test bit exactness of ITU test vectors) in frames of 160 samples. The file sample1.wav (input waveform file) would be analyzed and G726 A-law output would be stored in the sampleout.wav file.

```
C:\> testcodec eg7616alf160 sample1.wav dg7616alf160 sampleout.wav
```

Note : In this case, the μ-law values are stored as 16-bit integers in the wave file though the value of each sample is not greater than 255 (8-bit A-law).

5 Input and Output File Format

Simulation program input and output files can be raw header-less speech files or in .wav file format.

6 Intermediate Bitstream File Format

Simulation program intermediate bitstream files contain no header information, only packet data.

6.1 MELPe 600 bps Bitstream File Format

For 600 bps bitstream output, each packet contains 15 bytes; each byte has six (6) valid coefficient bits (upper 2 bits are zero). For a fixed-point simulation file output of N bytes length with no header information, actual valid data content can be expressed as:

$$\frac{N*90}{120}$$

Figure 1 below represents the fixed-point output bitstream file format valid.

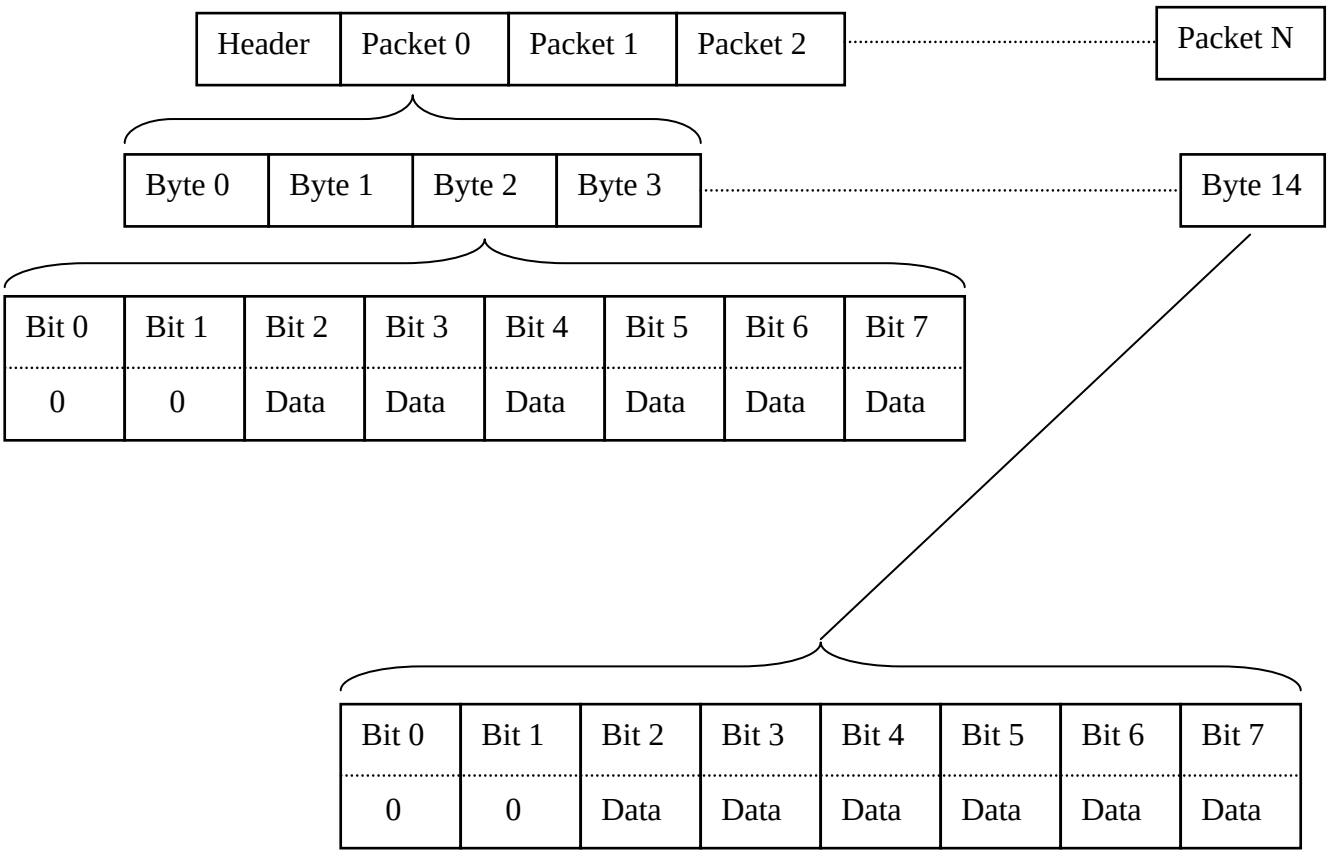


Figure 1 – Fixed-Point Output Bitstream File Format

As an example, consider a raw sound file called “fjd03.unx” has 37.919s duration, or 303352 16-bit samples, or 281 packets.

Therefore, calculating bitstream file length can be performed as follows:

1. Duration in seconds = 37.919 sec
2. Duration in samples = 37.919 x 8000 = 303352 samples (606704 bytes)
3. Number of packets = 303352/1080 = 280.881 packets = 281 packets (round up)
4. Physical file length is 4215 bytes (281 x 15)
5. 281 packets x 15 Bytes x 8 bits x 0.75 = 25290 bits
6. Total valid data in the file = 25290 bits, or 3161.5 bytes

Note that ratio of 4215 byte physical file size to 3161.5 byte file content size matches the 90/120 ratio given above.

6.2 MELPe 600 bps Bitstream File Format for 135 msec latency codec

For 600 bps bitstream output, each packet contains 14 bytes; each byte has six (6) valid coefficient bits (upper 2 bits are zero), except for the last byte in each packet, which has only 3 valid data bits. In other words, bytes 1-13 have 6 data bits, and byte 14 has 3 data bits. For a fixed-point simulation file output of N bytes length with no header information, actual valid data content can be expressed as:

N*81

112

Figure 2 below represents the fixed-point output bitstream file format valid.

As an example, consider a raw sound file called “fjd03.unx” has 37.919s duration, or 303352 16-bit samples, or 281 packets.

Therefore, calculating bitstream file length can be performed as follows:

7. Duration in seconds = 37.919 sec
8. Duration in samples = 37.919 x 8000 = 303352 samples (606704 bytes)
9. Number of packets = 303352/1080 = 280.881 packets = 281 packets (round up)
10. Physical file length is 3934 bytes (281 x 14)
11. 281 packets x 13 Bytes x 8 bits x 0.75 = 21918 bits
12. 281 packets x 3 bits = 843 bits
13. Total valid data in the file = 22761 bits, or 2846 bytes

Note that ratio of 3934 byte physical file size to 2846 byte file content size matches the 81/112 ratio given above.

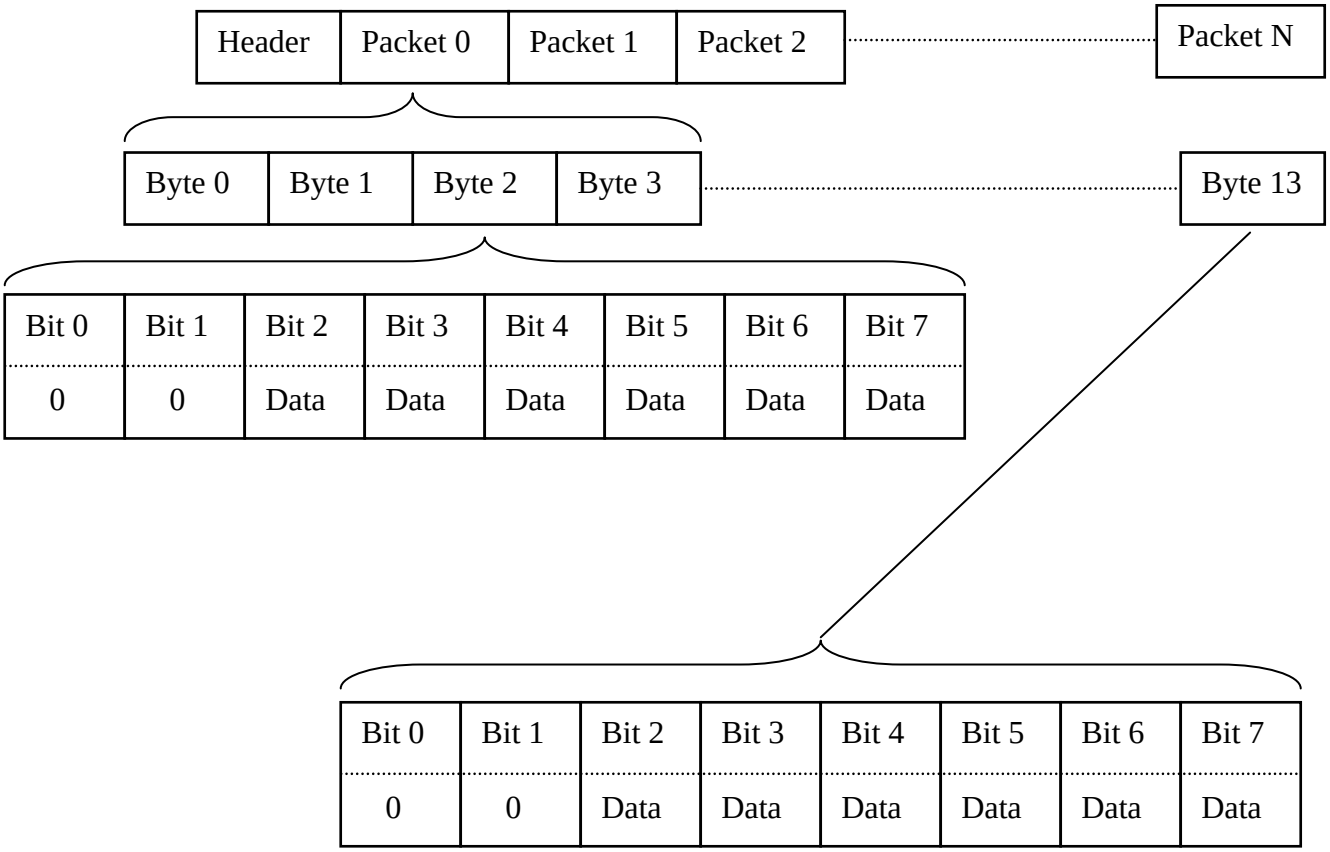


Figure 2 – Fixed-Point Output Bitstream File Format

7 Framing and Timing

The 600 bps MELPe-Plus codec has analysis module structure similar to 1200 bps MELPe codec, although frame and packet structure differs substantially.

A block, or “Superframe”, structure consisting of six consecutive subframes has been adopted to quantize transmitted parameters efficiently for 600 bps rate.

Basic algorithm parameters are as follows.

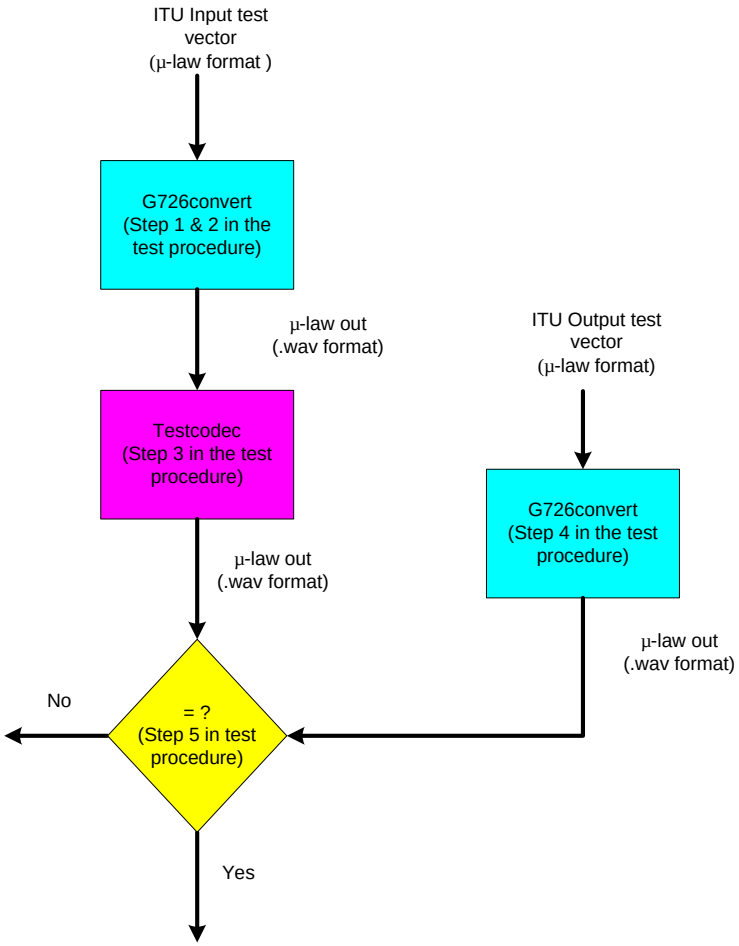
Superframe (Time Domain)		**Subframe**		**Bitstream (Packet Domain)**	
Samples	Msec	Samples	Msec	Bits	Msec
1200	150	200	25	90	150
1080	135	180	22.5	81	135

Table 3 - Framing and Timing for MELPe-Plus 600 bps Codec

8 G726 Bit Exactness Procedure

8.1 Procedure to test ITU G726 test vectors bit exactness

The following figure shows a flow chart of the G726 bit exactness test procedure:


ITU Input test vector (μ-law format)
G726convert (Step 1 & 2 in the test procedure)
μ-law out (.wav format)
Testcodec (Step 3 in the test procedure)
μ-law out (.wav format)
ITU Output test vector (μ-law format)
G726convert (Step 4 in the test procedure)
μ-law out (.wav format)
= ? (Step 5 in test procedure)
No
Yes

Below is a series of steps that execute the above procedure.

1) Change the file extension of the ITU input test vector “NRM.M” (it is a μ-law file in ASCII hex format) to “NRM.bin”. The command line is

```
C:/> copy nrm.m nrm.bin
```

2) Run the input test vector “NRM.bin” through *g726convert.exe*. The command line is

```
C:/> g726test nrm.bin nrmmulaw.wav
```

g726convert will convert the ASCII hex file to μ-law output as a wave file. Each value in the file is a *short int* (16 bits long) representing a μ-law sample (the value is not greater than 255). This is not a standard .wav format and would not be recognized correctly by media players.

3) Run the *testcodec* simulation program for G726 algorithm using “nrmmulaw.wav” as input. The command line is

```
C:/> testcodec eg7616muf160 nrmmulaw.wav dg7616muf160 nrmmuout.wav
```

“nrmmuout.wav” can be viewed by standard software though the values are not greater than 255, but would not be properly recognized by a media player.

4) Convert ITU 16 kbps output test vector file “RN16FM.O” from ASCII hex format to binary μ-law values for comparison. The command line is

```
C:/> g726convert rn16fm.o rn16fmo.wav
```

The values of the output wave file are in μ-law i.e. it is a wave file which has 16-bit integers whose values are not greater than 255. This is not a standard .wav format and would not be recognized correctly by media players.

5) Compare the two files “nrmmuout.wav” and “rn16fmo.wav” in MATLAB for bit exactness.

Notes

1) Length of “nrmmuout.wav” may be longer than that of the input file “NRM.M” because of a small amount of excess output data due to frame-based operation in the *testcodec* simulation and *g726convert* programs. So, when comparing “nrmmuout.bin” and “rn16fmo.wav”, consider only the number of values in the test vector in “nrmmuout.bin” disregarding any extra values.

2) The procedure for bit exactness does not cover all ITU test cases, for example intermediate encoder output.

9 Packet Bit Allocation

9.1 MELPe Packet Bit Allocation for 600 bps for 150 msec latency

Table 4 below shows overall packet bit allocation for the 600 bps codec for 150 msec latency. Note that one packet is equivalent to one Superframe (see section 7 above, Framing and Timing).

Parameter	**Bits**
Pitch	16
LSF	53
Gain	14
Bandpass Voicing	6
Synchronization	1
Total	**90**

Table 4 – Superframe Bit Allocation for MELPe-Plus 600 bps Codec

9.2 MELPe Packet Bit Allocation for 600 bps for 135 msec latency

Table 5 below shows overall packet bit allocation for the 600 bps codec for 135 msec latency. Note that one packet is equivalent to one Superframe (see section 7 above, Framing and Timing).

Parameter	**Bits**
Pitch	12
LSF	51
Gain	11
Bandpass Voicing	6
Synchronization	1
Total	**81**

Table 5 – Superframe Bit Allocation for MELPe-Plus 600 bps Codec